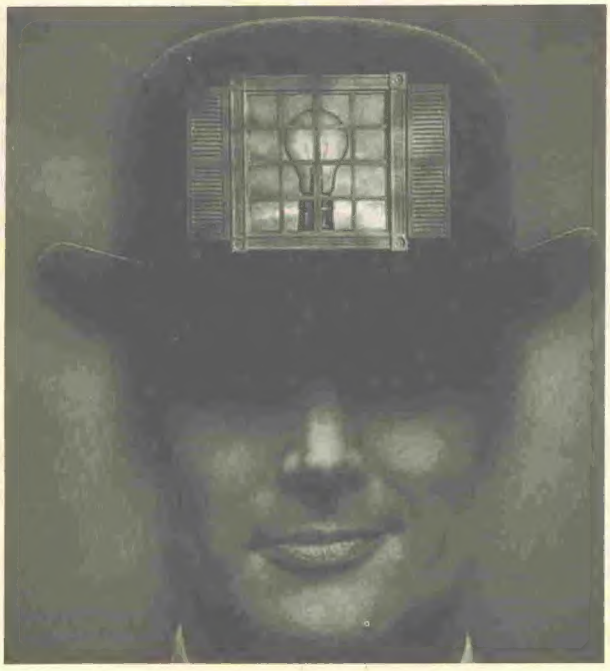
สุทธิศักดิ์ พงศ์ธนาพาณิช

Advanced BASIC Programming Technique ตอน กลุ่มฟังก์ชัน Rectangle และ Point ของวินโดวส์ 95



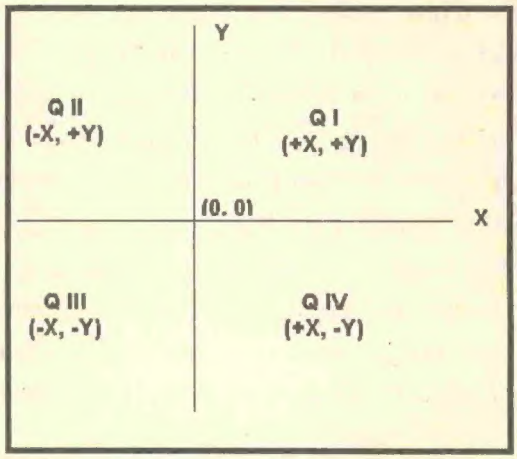
กลุ่มของฟังก์ชันพื้นฐานของวินโดวส์ 95 ซึ่งช่วยในการแสดงผลหรือประมวลผลด้านตรรกะของสี่เหลี่ยมและจุด

สำหรับเนื้อหาในตอนนี้ ผู้เขียนก็จะขออธิบายถึงกลุ่มฟังก์ชันพื้นฐานของวินโดวส์ 95 ซึ่งช่วยในการแสดงผลหรือประมวลผลด้านตรรกะของสี่เหลี่ยม (Rectangle) และจุด (Point) โดยที่ฟังก์ชันในกลุ่มนี้จะถูกนำไปใช้งานบ่อยที่สุดกับแอพพลิเคชันประเภทวาดกราฟิก เช่น Paint, CorelDaw เป็นต้น เพราะฟังก์ชันในกลุ่มนี้ช่วยในการประมวลผลตามเงื่อนไขด้านตรรกะ เช่น Intersection, Subtract หรือ Union เป็นต้น ระหว่างสี่เหลี่ยมสองรูป ซึ่งช่วยให้ผู้อ่านสามารถที่จะทราบถึงขอบเขตของสี่เหลี่ยมที่เป็นไปตามเงื่อนไขด้านตรรกะข้างต้น

ระบบแกนและข้อมูลโครงสร้าง

ระบบแกนตามแนวคิดด้านคณิตศาสตร์นั้น จะเป็นไปตามระบบแกนคาร์ทีเซียน (cartesian) ซึ่งแกนนอน (X) จะตัดกับแกนตั้ง (Y) ที่จุดกำเนิด (0,0) ดังนั้นในระบบแกนคาร์ทีเซียนจึงประกอบด้วยพื้นที่ 4 ส่วน (quardant) ดังในรูปที่ 1

และการแสดงผลก็จะเป็นในลักษณะของคู่ลำดับ (X,Y) เช่น (10, 100) จะเป็นตำแหน่งของคู่ลำดับ X=10 และ Y=100 ซึ่งจะเป็นตำแหน่งใน QI ของรูปที่ 1 เป็นต้น แต่สำหรับระบบแกนของเครื่องคอมพิวเตอร์โดยทั่วไป (ระบบแกนของจอภาพ) นั้นจะแตกต่างจากระบบแกนคาร์ทีเซียนพอสมควร เพราะระบบแกนของจอภาพจะมีจุดกำเนิด (0,0) ที่มุมบนซ้ายของจอภาพแสดงผล และแกน X, Y จะมีค่าเป็นบวกโดยมีทิศทางไปทางขวาและลงล่างตามลำดับ ดังนั้นระบบแกนของจอภาพจึง

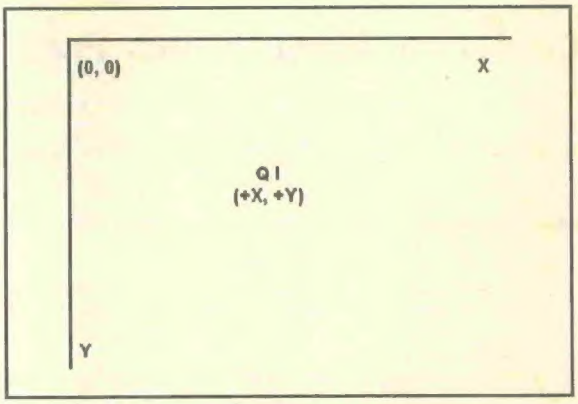


รูปที่ 1 แสดงระบบแกนคาร์ทีเซียน

เทียบเท่ากับ QI ของระบบแกนคาร์ทีเซียนนั่นเอง (แต่จะต้องกลับทิศทางของแกน Y ลงข้างล่าง) ดังในรูปที่ 2

สาเหตุที่ระบบแกนของจอภาพมีค่า X และ Y เป็นบวกอย่างเดียว ก็เพื่อให้ตัวเลขที่กำหนดตำแหน่งบนจอภาพนั้น มีค่าเป็นบวกซึ่งจะทำให้การเขียนโปรแกรมง่ายยิ่งขึ้น เพราะไม่ต้องกังวลเกี่ยวกับเครื่องหมายของค่า X หรือ Y ดังนั้นถ้าหากผู้อ่านต้องการแปลงรูปสี่เหลี่ยมจากระบบแกนคาร์ทีเซียน มายังระบบแกนของจอภาพ ก็ต้องคำนึงถึงความแตกต่างของระบบแกนด้วยเช่นกัน

สำหรับข้อมูลโครงสร้างที่ใช้กับกลุ่มของฟังก์ชัน Rectangle และ Point นั้นจะประกอบด้วย 2 ข้อมูล ดังต่อไปนี้



รูปที่ 2 แสดงระบบแกนของจอภาพ

ข้อมูลโครงสร้าง RECT

เป็นข้อมูลโครงสร้างที่ใช้กับกลุ่มของฟังก์ชัน Rectangle สำหรับการกำหนดตำแหน่งขอบเขตของสี่เหลี่ยม ซึ่งประกอบด้วยสมาชิกดังต่อไปนี้

```
Type Rect
    Left    As Long
    Top     As Long
    Right   As Long
    Bottom  As Long
End Type
```

Left และ Top สำหรับกำหนดตำแหน่งมุมแรกของสี่เหลี่ยม (โดยปกติจะหมายถึงมุมบนซ้ายของสี่เหลี่ยม)

Right และ Bottom สำหรับกำหนดขนาดในแนวนอนและขนาดในแนวตั้งของสี่เหลี่ยมตามลำดับ

โดยที่หน่วยที่ใช้ในการบอกตำแหน่งหรือขนาดของสี่เหลี่ยมนั้น จะแปรเปลี่ยนไปตามหน่วยของออบเจ็กต์ที่ใช้วาดกราฟิก เช่น Form หรือ PictureBox เป็นต้น สำหรับ Visual Basic ผู้อ่านสามารถกำหนดหน่วยวัดขนาดของออบเจ็กต์ได้ด้วยคุณสมบัติ ScaleMode ของออบเจ็กต์นั้น ๆ และสำหรับชนิดของข้อมูลสำหรับสมาชิกของโครงสร้าง Rect จะเป็นข้อมูลชนิด Long สำหรับระบบปฏิบัติการวินโดวส์ 95

ก่อนที่ผู้เขียนจะข้ามไปยังหัวข้อถัดไป ก็จะขออธิบายเพิ่มเติมเกี่ยวกับการจัดเก็บค่าตัวเลขของโครงสร้าง Rect เสียก่อน ทั้งนี้เนื่องจากถ้าหากผู้อ่านกำหนดให้สมาชิกทั้ง 4 ของข้อมูลโครงสร้าง Rect มีค่าเท่ากับ 0 หรือมีค่าเท่ากัน เช่น 100 เป็นต้น ก็ไม่ได้หมายความว่าเป็นการแสดงผลจุดที่ตำแหน่งตามที่กำหนด แต่สำหรับวินโดวส์แล้วการกำหนดค่าให้สมาชิกทั้ง 4 ของข้อมูลโครงสร้าง Rect ให้เท่ากับ 0 ก็จะหมายถึงสี่เหลี่ยมว่างเปล่า (empty rectangle) ซึ่งจะไม่มีการแสดงผลใด ๆ ทั้งสิ้น ดังนั้นถ้าหากผู้อ่านต้องการกำหนดให้สี่เหลี่ยมมีขนาดความกว้างและความสูงเท่ากับ 1 หน่วย (แสดงผลเป็นจุด) ก็ต้องกำหนดให้ค่าของสมาชิก Right และ Bottom มีค่ามากกว่าค่าของสมาชิก Left และ Top อยู่ 1 หน่วย ตามลำดับ เช่น

```
Dim Rect1 As Rect

Rect1.Left = 10
Rect1.Top = 10
Rect1.Right = 11
Rect1.Bottom = 11
```

ดังนั้นผู้อ่านจึงสามารถคำนวณขนาดของสี่เหลี่ยมจากข้อมูลโครงสร้าง Rect ด้วยสูตรดังต่อไปนี้

```
Width& = Rect1.Right - Rect1.Left
Height& = Rect1.Bottom - Rect1.Top
```

ข้อมูลโครงสร้าง PointAPI

เป็นข้อมูลโครงสร้างที่ใช้กับกลุ่มของฟังก์ชัน Point สำหรับการกำหนดตำแหน่งของจุด (ในความเป็นจริงข้อมูลโครงสร้าง PointAPI นั้นจะชื่อ Point แต่ในที่นี้ผู้เขียนกำหนดให้ชื่อ PointAPI ก็เพื่อไม่ให้สับสนกับวิธี Point ของออบเจ็กต์ของ Visual Basic) ซึ่งประกอบด้วยสมาชิกดังต่อไปนี้

```
Type PointAPI
    X   As Long
    Y   As Long
End Type
```

X และ Y สำหรับกำหนดตำแหน่งของโคออร์ดิเนต X และ Y ของระบบแกนจอภาพ ตามลำดับ

สำหรับในหัวข้อถัดไป ผู้เขียนก็จะขออธิบายถึงรายละเอียดเกี่ยวกับฟังก์ชันที่เกี่ยวข้องกับข้อมูลโครงสร้างทั้งสอง

กลุ่มของฟังก์ชัน Rectangle และ Point

สำหรับกลุ่มของฟังก์ชัน Rectangle และ Point ที่ผู้เขียนจะอธิบายในบทความตอนนี้ ก็จะเน้นเฉพาะฟังก์ชันที่นำมาใช้ในโปรเจ็กต์ตัวอย่าง RectAPI เท่านั้น แต่ถ้าหากมีฟังก์ชันใดที่ไม่ถูกนำมาใช้งานก็จะไม่ขอกล่าวรายละเอียดในบทความตอนนี้ สำหรับกลุ่มของฟังก์ชัน Rectangle และ Point ที่จะกล่าวรายละเอียดมีดังต่อไปนี้

```
Declare Function InflateRect Lib "user32" (lpRect As Rect, _
        ByVal X As Long, ByVal Y As Long) As Long
Private Declare Function IntersectRect Lib "user32" (lpDestRect As Rect, _
        lpSrc1Rect As Rect, lpSrc2Rect As Rect) As Long
Private Declare Function IsRectEmpty Lib "user32" (lpRect As Rect) As Long
Private Declare Function OffsetRect Lib "user32" (lpRect As Rect, _
        ByVal X As Long, ByVal Y As Long) As Long
Private Declare Function PtInRect Lib "user32" (lpRect As Rect, _
        ByVal ptx As Long, ByVal pty As Long) As Long
Private Declare Function SetRect Lib "user32" (lpRect As Rect, _
        ByVal X1 As Long, ByVal Y1 As Long, ByVal X2 As Long, ByVal Y2 As Long) As Long
Private Declare Function SetRectEmpty Lib "user32" (lpRect As Rect) As Long
Private Declare Function SubtractRect Lib "user32" (lprcDst As Rect, _
        lprcSrc1 As Rect, lprcSrc2 As Rect) As Long
Private Declare Function UnionRect Lib "user32" (lpDestRect As Rect, _
        lpSrc1Rect As Rect, lpSrc2Rect As Rect) As Long
```

InflateRect

การประกาศ

```
Declare Function InflateRect Lib "user32" (lpRect As Rect, _
ByVal X As Long, ByVal Y As Long) As Long
```

หน้าที่ ปรับเปลี่ยนขนาดความกว้างหรือความสูงของสี่เหลี่ยม

พารามิเตอร์
- lpRect ข้อมูลโครงสร้าง Rect ที่จัดเก็บสี่เหลี่ยม
- X ค่าตัวเลขจำนวนเต็ม long สำหรับเพิ่ม (X > 0) หรือลดขนาด (X < 0) ความกว้างของสี่เหลี่ยม
- Y ค่าตัวเลขจำนวนเต็ม long สำหรับเพิ่ม (Y > 0) หรือลดขนาด (Y < 0) ความสูงของสี่เหลี่ยม

ตัวอย่างการใช้

```
Xval& = 10   ' Increase width 10 unit
Yval& = -5   ' Decrease height 5 unit
Success& = InflateRect(tmpRect, Xval&, Yval&)
```

ค่าที่ส่งกลับ
- <> 0 หมายถึง ฟังก์ชันทำงานสำเร็จ
- = 0 หมายถึง ฟังก์ชันทำงานไม่สำเร็จ

หมายเหตุ ขนาดของสี่เหลี่ยมที่ถูกปรับเปลี่ยนด้วยฟังก์ชันนี้ จะเป็นค่า 2 เท่าของพารามิเตอร์ X และ Y เช่น ถ้าหากผู้อ่านกำหนดให้ X = 10 สี่เหลี่ยมก็จะถูกเพิ่มขนาดความกว้างข้างล่าง 10 หน่วย หรือสี่เหลี่ยมก็จะถูกเพิ่มขนาดความกว้างทั้งสิ้น 20 หน่วยนั่นเอง

IntersectRect

การประกาศ

```
Private Declare Function IntersectRect Lib "user32" _
(lpDestRect As Rect, lpSrc1Rect As Rect,lpSrc2Rect As Rect) As Long
```

หน้าที่ คำนวณพื้นที่ที่ซ้อนทับกัน (intersection) ของสี่เหลี่ยม 2 รูป

พารามิเตอร์
- lpDestRect ข้อมูลโครงสร้าง Rect ที่ใช้จัดเก็บผลลัพธ์โคออร์ดิเนตของสี่เหลี่ยม ที่ได้จากการคำนวณด้านตรรกะ intersection
- lpSrc1Rect ข้อมูลโครงสร้าง Rect ของสี่เหลี่ยมต้นทาง ที่ใช้คำนวณด้านตรรกะ intersection
- lpSrc2Rect ข้อมูลโครงสร้าง Rect ของสี่เหลี่ยมปลายทาง ที่ใช้คำนวณด้านตรรกะ intersection ร่วมกับสี่เหลี่ยมต้นทาง

ตัวอย่างการใช้

```
Success& = IntersectRect(RectIntersect, Rect1, Rect2)
' Paint intersection area
Picture1.Line (RectIntersect.Left, RectIntersect.Top)- _
(RectIntersect.Right, RectIntersect.Bottom), QBColor(8), BF
```

→

ค่าที่ส่งกลับ <> 0 หมายถึง ฟังก์ชันทำงานสำเร็จ
= 0 หมายถึง ฟังก์ชันทำงานไม่สำเร็จ

หมายเหตุ ถ้าหากสี่เหลี่ยมต้นทางและสี่เหลี่ยมปลายทางไม่มีส่วนใดที่ซ้อนทับกันเลย พารามิเตอร์ lpDestRect ก็จะมีค่าเท่ากับสี่เหลี่ยมว่าง (empty rectangle)

■

IsRectEmpty

การประกาศ

```
Private Declare Function IsRectEmpty Lib "user32" _
(lpRect As Rect) As Long
```

หน้าที่ ตรวจสอบสี่เหลี่ยมที่กำหนด เป็นสี่เหลี่ยมว่างหรือไม่

พารามิเตอร์ lpRect ข้อมูลโครงสร้าง Rect ที่จัดเก็บสี่เหลี่ยม ที่ต้องการทดสอบ

ตัวอย่างการใช้

```
' Empty Rect1 if exist
If IsRectEmpty(Rect1) = 0 Then
      Success& = SetRectEmpty(Rect1)
End If
```

ค่าที่ส่งกลับ <> 0 หมายถึง สี่เหลี่ยมที่กำหนด เป็นสี่เหลี่ยมว่าง
= 0 หมายถึง สี่เหลี่ยมที่กำหนด ไม่เป็นสี่เหลี่ยมว่าง

หมายเหตุ สี่เหลี่ยมว่าง หมายถึง สี่เหลี่ยมที่สมาชิกทั้ง 4 ของข้อมูลโครงสร้าง Rect มีค่าเท่ากับ 0

OffsetRect

การประกาศ

```
Private Declare Function OffsetRect Lib "user32" _
(lpRect As Rect, ByVal X As Long, ByVal Y As Long) AsLong
```

หน้าที่ เคลื่อนย้ายสี่เหลี่ยมไปยังตำแหน่ง X หรือ Y ที่กำหนด

พารามิเตอร์ lpRect ข้อมูลโครงสร้าง Rect ที่จัดเก็บสี่เหลี่ยม
X ค่าตัวเลขจำนวนเต็ม long สำหรับย้ายสี่เหลี่ยมไปทางขวา (X > 0) หรือทางซ้าย (X < 0)
Y ค่าตัวเลขจำนวนเต็ม long สำหรับย้ายสี่เหลี่ยมลงล่าง (Y > 0) หรือขึ้นบน (Y < 0)

ตัวอย่างการใช้

```
' Empty Rect1 if exist
If IsRectEmpty(Rect1), = 0 Then
  Success& = SetRectEmpty(Rect1)
End If
```

ค่าที่ส่งกลับ <> 0 หมายถึง ฟังก์ชันทำงานสำเร็จ
= 0 หมายถึง ฟังก์ชันทำงานไม่สำเร็จ

หมายเหตุ สำหรับทิศทางที่ทำการเคลื่อนย้ายสี่เหลี่ยมนั้น อาจจะแตกต่างจากที่กล่าวมาแล้วก็ได้ ทั้งนี้ขึ้นกับระบบแกนที่กำหนดด้วยเช่นกัน

PtInRect

การประกาศ

```
Private Declare Function PtInRect Lib "user32" (lpRect As Rect, _
ByVal ptx As Long, ByVal pty As Long) As Long
```

หน้าที่ ตรวจสอบตำแหน่งของจุดที่กำหนด ว่าอยู่ในพื้นที่ของสี่เหลี่ยมหรือไม่

พารามิเตอร์ lpRect ข้อมูลโครงสร้าง Rect ที่จัดเก็บสี่เหลี่ยม
ptx ค่าตัวเลขจำนวนเต็ม long ที่กำหนดตำแหน่งโคออร์ดิเนต X ของจุด
pty ค่าตัวเลขจำนวนเต็ม long ที่กำหนดตำแหน่งโคออร์ดิเนต Y ของจุด

→

ตัวอย่างการใช้

```
' read Point location
Xval& = EndPoint.X
Yval& = EndPoint.Y

' check Point location for each case
If PtInRect(Rect1, Xval&, Yval&) Then p$ = "[R1] "
```

ค่าที่ส่งกลับ
<> 0 หมายถึง ตำแหน่งของจุดที่กำหนด อยู่ในพื้นที่ของสี่เหลี่ยม
= 0 หมายถึง ตำแหน่งของจุดที่กำหนด ไม่อยู่ในพื้นที่ของสี่เหลี่ยม

หมายเหตุ ไม่มี ■

SetRect

การประกาศ

```
Private Declare Function SetRect Lib "user32" (lpRect As Rect, _
ByVal X1 As Long, ByVal Y1 As Long, ByVal X2 As Long,ByVal Y2 As Long) As Long
```

หน้าที่ กำหนดตำแหน่งโคออร์ดิเนตทั้ง 4 ให้กับสี่เหลี่ยม

พารามิเตอร์
lpRect ข้อมูลโครงสร้าง Rect ของสี่เหลี่ยมที่ต้องการกำหนด
X1 ค่าตัวเลขจำนวนเต็ม long ที่กำหนดตำแหน่งโคออร์ดิเนตสำหรับสมาชิก Left ของข้อมูลโครงสร้าง Rect
Y1 ค่าตัวเลขจำนวนเต็ม long ที่กำหนดตำแหน่งโคออร์ดิเนตสำหรับสมาชิก Top ของข้อมูลโครงสร้าง Rect
X2 ค่าตัวเลขจำนวนเต็ม long ที่กำหนดตำแหน่งโคออร์ดิเนตสำหรับสมาชิก Right ของข้อมูลโครงสร้าง Rect
Y2 ค่าตัวเลขจำนวนเต็ม long ที่กำหนดตำแหน่งโคออร์ดิเนตสำหรับสมาชิก Bottom ของข้อมูลโครงสร้าง Rect

ตัวอย่างการใช้

```
SetRect Rect1, StartPoint.X, StartPoint.Y, EndPoint.X, EndPoint.Y
Picture1.Line (StartPoint.X, StartPoint.Y)-(EndPoint.X, EndPoint.Y), , B
```

ค่าที่ส่งกลับ
<> 0 หมายถึง ตำแหน่งของจุดที่กำหนด อยู่ในพื้นที่ของสี่เหลี่ยม
= 0 หมายถึง ตำแหน่งของจุดที่กำหนด ไม่อยู่ในพื้นที่ของสี่เหลี่ยม

หมายเหตุ สำหรับฟังก์ชันนี้ จะเทียบเท่ากับการกำหนดค่าให้กับสมาชิกทั้ง 4 ของข้อมูลโครงสร้าง Rect ด้วยโค้ดต่อไปนี้

Rect2.Left = X1 : Rect2.Top = Y1
Rect2.Right = X2: Rect2.Bottom = Y2

SetRectEmpty

การประกาศ

```
Private Declare Function SetRectEmpty Lib "user32" _
(lpRect As Rect) As Long
```

หน้าที่ กำหนดให้สี่เหลี่ยม เป็นสี่เหลี่ยมว่าง

พารามิเตอร์ lpRect ข้อมูลโครงสร้าง Rect ของสี่เหลี่ยมที่ต้องการกำหนดให้เป็นสี่เหลี่ยมว่าง

ตัวอย่างการใช้

```
' Empty Rect1 if exist
If IsRectEmpty(Rect1) = 0 Then
      Success& = SetRectEmpty(Rect1)
End If
```

ค่าที่ส่งกลับ
<> 0 หมายถึง ฟังก์ชันทำงานสำเร็จ
= 0 หมายถึง ฟังก์ชันทำงานไม่สำเร็จ

หมายเหตุ สำหรับฟังก์ชันนี้ จะเทียบเท่ากับการกำหนดค่าให้กับสมาชิกทั้ง 4 ของข้อมูลโครงสร้าง ให้มีค่าเท่ากับ 0 ด้วยโค้ดต่อไปนี้

Rect2.Left = 0 : Rect2.Top = 0
Rect2.Right = 0: Rect2.Bottom = 0

SubtractRect

การประกาศ

```
Private Declare Function SubtractRect Lib "user32" _
(lprcDst As Rect, lprcSrc1 As Rect, lprcSrc2 As Rect) As Long
```

หน้าที่ คำนวณพื้นที่ที่เกิดจากการลบสี่เหลี่ยม lprcSrc2 ออกจาก lprcSrc1

พารามิเตอร์
- lprcDst ข้อมูลโครงสร้าง Rect ที่ใช้จัดเก็บผลลัพธ์โคออร์ดิเนตของสี่เหลี่ยม ที่ได้จากการคำนวณด้านตรรกะ subtraction
- lprcSrc1 ข้อมูลโครงสร้าง Rect ของสี่เหลี่ยมตัวตั้ง สำหรับคำนวณด้านตรรกะ subtraction
- lprcSrc2 ข้อมูลโครงสร้าง Rect ของสี่เหลี่ยมตัวลบ สำหรับคำนวณด้านตรรกะ subtraction ร่วมกับสี่เหลี่ยมตัวตั้ง

ตัวอย่างการใช้

```
Success& = SubtractRect(RectSubtract, Rect1, Rect2)
Picture1.Line (RectSubtract.Left, RectSubtract.Top)- _
(RectSubtract.Right, RectSubtract.Bottom), QBColor(4), BF
```

ค่าที่ส่งกลับ
- <> 0 หมายถึง ฟังก์ชันทำงานสำเร็จ
- = 0 หมายถึง ฟังก์ชันทำงานไม่สำเร็จ

หมายเหตุ เนื่องจากฟังก์ชันนี้ จะดำเนินการคำนวณด้านตรรกะ subtraction ระหว่างสี่เหลี่ยมทั้งสอง ดังนั้นผู้อ่านสามารถลบสี่เหลี่ยมแรกออกจากสี่เหลี่ยมที่สอง หรือสี่เหลี่ยมที่สองออกจากสี่เหลี่ยมแรก ก็ให้สลับพารามิเตอร์ ต่อไปนี้

```
' Subtraction : Rect1 - Rect2
Success& = SubtractRect(RectSubtract, Rect1, Rect2)
' Subtraction : Rect2 - Rect1
Success& = SubtractRect(RectSubtract, Rect2, Rect1)
```

UnionRect

การประกาศ

```
Private Declare Function UnionRect Lib "user32" _
(lpDestRect As Rect, lpSrc1Rect As Rect, lpSrc2Rect As Rect) As Long
```

หน้าที่ คำนวณพื้นที่ที่เกิดจากการรวมสี่เหลี่ยม lpSrc1Rect เข้ากับ lpSrc2Rect ด้วยการคำนวณด้านตรรกะ union

พารามิเตอร์
- lpDestRect ข้อมูลโครงสร้าง Rect ที่ใช้จัดเก็บผลลัพธ์โคออร์ดิเนตของสี่เหลี่ยม ที่ได้จากการคำนวณด้านตรรกะ union
- lpSrc1Rect ข้อมูลโครงสร้าง Rect ของสี่เหลี่ยมแรก สำหรับคำนวณด้านตรรกะ union
- lpSrc2Rect ข้อมูลโครงสร้าง Rect ของสี่เหลี่ยมที่สอง สำหรับคำนวณด้านตรรกะ union ร่วมกับสี่เหลี่ยมแรก

ตัวอย่างการใช้

```
Success& = UnionRect(RectUnion, Rect1, Rect2)
Picture1.Line (RectUnion.Left, RectUnion.Top)- _
(RectUnion.Right, RectUnion.Bottom), QBColor(5), BF
```

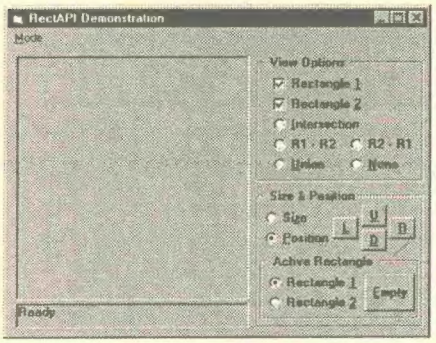
ค่าที่ส่งกลับ
- <> 0 หมายถึง ฟังก์ชันทำงานสำเร็จ
- = 0 หมายถึง ฟังก์ชันทำงานไม่สำเร็จ

หมายเหตุ ไม่มี

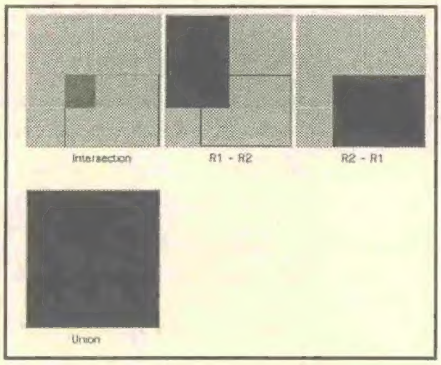
โปรเจ็กต์ตัวอย่าง RectAPI.vbp

โปรเจ็กต์ตัวอย่าง RectAPI.vbp เป็นโปรเจ็กต์ที่แสดงวิธีการใช้กลุ่มของฟังก์ชัน Rectangle หรือ Point ในการประมวลผลสี่เหลี่ยม โดยจะนำฟังก์ชันทั้งหมดที่ได้กล่าวรายละเอียดในข้างต้น มาใช้ประมวลผลสี่เหลี่ยมในรูปแบบต่าง ๆ ซึ่งผู้อ่านสามารถดูผลลัพธ์ของฟังก์ชันเหล่านี้ได้จากกราฟิกที่แสดงผลในคอนโทรล Picture1 ของฟอร์ม RectAPI นี้เอง โดยเมื่อผู้อ่านรันโปรเจ็กต์ก็จะปรากฏฟอร์ม RectAPI ดังในรูปที่ 3 จากนั้นก็ให้ผู้อ่านทำการสร้างสี่เหลี่ยม 2 รูป (Rect1 และ Rect2) โดยดำเนินตามขั้นตอนต่อไปนี้

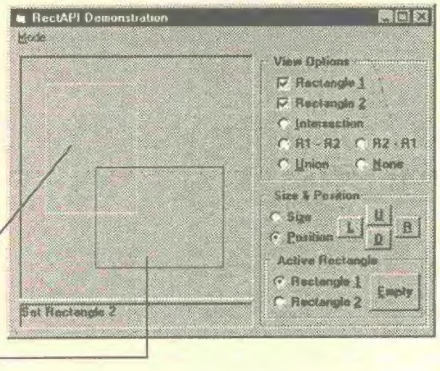
1. เลือกคำสั่ง Set Rect1 จากเมนู Mode เพื่อสร้างสี่เหลี่ยมต้นทาง
2. คลิกเม้าส์ภายในพื้นที่ของคอนโทรล Picture1 และลากเม้าส์เพื่อให้ได้สี่เหลี่ยมต้นทางที่มี
3. ขนาดตามที่ต้องการ จากนั้นก็ปล่อยเม้าส์ สี่เหลี่ยมต้นทางก็จะถูกสร้างด้วยสีที่กำหนด
4. เลือกคำสั่ง Set Rect2 จากเมนู Mode เพื่อสร้างสี่เหลี่ยมปลายทาง
5. คลิกเม้าส์ภายในพื้นที่ของคอนโทรล Picture1 และลากเม้าส์เพื่อให้ได้สี่เหลี่ยมปลายทางที่มีขนาดตามที่ต้องการ จากนั้นก็ปล่อยเม้าส์ สี่เหลี่ยมปลายทางก็จะถูกสร้างด้วยสีที่กำหนด



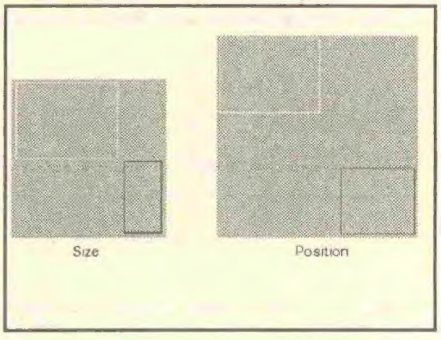
รูปที่ 3
แสดงหน้าต่างหลักของโปรเจ็กต์ RectAPI



รูปที่ 5
แสดงพื้นที่ของสี่เหลี่ยมที่ได้จากการประมวลผลด้านตรรกะ



รูปที่ 4
แสดงสี่เหลี่ยมต้นทางและสี่เหลี่ยมปลายทางในคอนโทรล Picture1



รูปที่ 6
แสดงการปรับขนาดหรือเคลื่อนย้ายสี่เหลี่ยม

จากนั้นก็ขอให้ผู้อ่านทดลองคลิกที่ปุ่ม Option ต่าง ๆ ในส่วนของ View Options เพื่อดูพื้นที่ของสี่เหลี่ยมที่ได้จากการประมวลผลด้านตรรกะต่าง ๆ โดยในรูปที่ 5 แสดงให้เห็นถึงผลลัพธ์ที่ได้ โดยจะแสดงผลเป็นพื้นที่ระบายสี

สำหรับการปรับขนาดหรือเคลื่อนย้ายสี่เหลี่ยม ก็ให้ผู้อ่านคลิกที่ปุ่ม Option ต่าง ๆ ในส่วนของ Size&Position เพื่อดำเนินการ จากนั้นก็คลิกที่ปุ่มคำสั่งทั้งสี่ เพื่อปรับขนาดหรือเคลื่อนย้ายสี่เหลี่ยม โดยในรูปที่ 6 จะแสดงให้เห็นถึงผลลัพธ์การดำเนินการทั้งสอง โดยที่ผู้อ่านสามารถเลือกที่จะดำเนินการกับสี่เหลี่ยมต้นทาง (Rectangle 1) หรือสี่เหลี่ยมปลายทาง (Rectangle 2) โดยการกำหนดสี่เหลี่ยมแอคทีฟจากการเลือกปุ่ม Option จากส่วนของ Active Rectangle เช่นจากรูปที่ 4 ก็แสดงว่าสี่เหลี่ยมต้นทาง (Rectangle 1) เป็นสี่เหลี่ยมแอคทีฟ

ลิสติงที่ 1 ไฟล์โปรเจ็กต์ RectAPI.vbp

```
Form=RectAPI.frm
ProjWinSize=152,402,248,215
ProjWinShow=2
IconForm="RectAPI"
HelpFile=""
Title="RectAPI"
ExeName32="RectAPI.exe"
Path32="c:\"
Name="AdvVbProgRectAPI"
HelpContextID="0"
StartMode=0
VersionCompatible32="0"
VersionCompatible="0"
MajorVer=1
MinorVer=0
RevisionVer=0
AutoIncrementVer=0
ServerSupportFiles=0
VersionComments="Advanced Basic Programming Technique"
VersionCompanyName="SE-ED Co.,Ltd."
VersionFileDescription="Rect API Functions"
VersionLegalCopyright="Sutthisak Phongthanapanich"
VersionProductName="Microcomputer Magazine"
```

ส่วนในการลบทิ้งหรือกำหนดให้สี่เหลี่ยมเป็นสี่เหลี่ยมว่าง ก็ให้คลิกที่ปุ่มคำสั่ง Empty สี่เหลี่ยมแอคทีฟก็จะถูกลบทิ้งทันที

จากเนื้อหาที่กล่าวมาทั้งหมด ผู้อ่านก็คงจะเข้าใจวิธีการใช้ฟังก์ชันวินโดว์ API สำหรับกลุ่มของฟังก์ชัน Rectangle หรือ Point พอสมควร ซึ่งผู้อ่านคงจะสามารถนำไปใช้ประโยชน์กับการสร้างแอพพลิเคชันอื่น ๆ ได้ต่อไปนะครับ สำหรับในตอนหน้าผู้เขียนก็จะไปเจาะลึกถึงฟังก์ชันวินโดว์ API กลุ่มอื่น ๆ ที่สนับสนุนวินโดวส์ 32 บิตกันต่อไป ก็คอยติดตามกันต่อไปนะครับ พบกันใหม่ฉบับหน้าสวัสดีครับ

ลิสติงที่ 2
ไฟล์ฟอร์ม RectAPI.frm

```
VERSION 4.00
Begin VB.Form RectAPI
   Caption         =   "RectAPI Demonstration"
   ClientHeight    =   4245
   ClientLeft      =   1020
   ClientTop       =   1905
   ClientWidth     =   6180
   BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
      Name             =   "MS Sans Serif"
      Size             =   8.25
      Charset          =   222
      Weight           =   700
      Underline        =   0   'False
      Italic           =   0   'False
      Strikethrough    =   0   'False
   EndProperty
   ForeColor       =   &H80000008&
   Height          =   4935
   Left            =   960
   LinkMode        =   1  'Source
   LinkTopic       =   "Form1"
   LockControls    =   -1  'True
   ScaleHeight     =   4245
   ScaleWidth      =   6180
   Top             =   1275
   Width           =   6300
   Begin VB.Frame Frame2
      Caption          =   " Size && Position"
      ForeColor        =   &H00FF0000&
      Height           =   2055
      Left             =   3660
      TabIndex         =   8
      Top              =   2100
      Width            =   2475
      Begin VB.Frame Frame3
         Caption          =   " Active Rectangle "
         ForeColor        =   &H00FF0000&
         Height           =   975
         Left             =   60
         TabIndex         =   15
         Top              =   960
         Width            =   2355
         Begin VB.CommandButton Command2
            Caption          =   "&Empty"
            Height           =   555
            Left             =   1560
            TabIndex         =   18
            Top              =   300
            Width            =   735
         End
         Begin VB.OptionButton Option3
            Caption          =   "Rectangle &2"
            Height           =   255
            Index            =   1
            Left             =   120
            TabIndex         =   17
            Top              =   600
            Width            =   1395
         End
         Begin VB.OptionButton Option3
            Caption          =   "Rectangle &1"
            Height           =   255
            Index            =   0
            Left             =   120
            TabIndex         =   16
            Top              =   300
            Value            =   -1  'True
            Width            =   1395
         End
      End
      Begin VB.CommandButton Command1
         Caption          =   "&R"
```

→

ลิสติงที่ 2 (ต่อ)

```
      Height          =   315
      Index           =   3
      Left            =   1980
      TabIndex        =   14
      Top             =   420
      Width           =   375
   End
   Begin VB.CommandButton Command1
      Caption         =   "&L"
      Height          =   315
      Index           =   2
      Left            =   1140
      TabIndex        =   13
      Top             =   420
      Width           =   375
   End
   Begin VB.CommandButton Command1
      Caption         =   "&D"
      Height          =   315
      Index           =   1
      Left            =   1560
      TabIndex        =   12
      Top             =   600
      Width           =   375
   End
   Begin VB.CommandButton Command1
      Caption         =   "&U"
      Height          =   315
      Index           =   0
      Left            =   1560
      TabIndex        =   11
      Top             =   240
      Width           =   375
   End
   Begin VB.OptionButton Option2
      Caption         =   "&Position"
      Height          =   255
      Index           =   1
      Left            =   120
      TabIndex        =   10
      Top             =   600
      Value           =   -1  'True
      Width           =   975
   End
   Begin VB.OptionButton Option2
      Caption         =   "Si&ze"
      Height          =   255
      Index           =   0
      Left            =   120
      TabIndex        =   9
      Top             =   300
      Width           =   975
   End
End
Begin VB.Frame Frame1
   Caption         =   " View Options "
   ForeColor       =   &H00FF0000&
   Height          =   1875
   Left            =   3660
   TabIndex        =   2
   Top             =   120
   Width           =   2475
   Begin VB.OptionButton Option1
      Caption         =   "&None"
      Height          =   255
      Index           =   4
      Left            =   1380
      TabIndex        =   20
      Top             =   1500
      Width           =   1035
   End
   Begin VB.OptionButton Option1
      Caption         =   "&Union"
```

→

```
      Height          =   255
      Index           =   3
      Left            =   240
      TabIndex        =   19
      Top             =   1500
      Width           =   1035
   End
   Begin VB.OptionButton Option1
      Caption         =   "R2 - R1"
      Height          =   255
      Index           =   2
      Left            =   1380
      TabIndex        =   7
      Top             =   1200
      Width           =   1035
   End
   Begin VB.OptionButton Option1
      Caption         =   "R1 - R2"
      Height          =   255
      Index           =   1
      Left            =   240
      TabIndex        =   6
      Top             =   1200
      Width           =   1095
   End
   Begin VB.OptionButton Option1
      Caption         =   "&Intersection "
      Height          =   255
      Index           =   0
      Left            =   240
      TabIndex        =   5
      Top             =   900
      Width           =   1400
   End
   Begin VB.CheckBox Check1
      Caption         =   "Rectangle &2"
      Height          =   375
      Index           =   1
      Left            =   240
      TabIndex        =   4
      Top             =   540
      Value           =   1  'Checked
      Width           =   1750
   End
   Begin VB.CheckBox Check1
      Caption         =   "Rectangle &1"
      Height          =   375
      Index           =   0
      Left            =   240
      TabIndex        =   3
      Top             =   240
      Value           =   1  'Checked
      Width           =   1750
   End
End
Begin VB.PictureBox Picture1
   Height          =   3615
   Left            =   120
   ScaleHeight     =   237
   ScaleMode       =   3  'Pixel
   ScaleWidth      =   225
   TabIndex        =   0
   Top             =   120
   Width           =   3435
End
Begin VB.Label Label1
   BorderStyle     =   1  'Fixed Single
   Caption         =   "Ready"
   Height          =   375
   Left            =   120
   TabIndex        =   1
   Top             =   3780
   Width           =   3435
End
```

→

ลิสติงที่ 2 (ต่อ)

```
   Begin VB.Menu mnuMode
      Caption         =   "&Mode"
      Begin VB.Menu mnuModeItem
         Caption          =   "&Point"
         Index            =   0
      End
      Begin VB.Menu mnuModeItem
         Caption          =   "Set Rect &1"
         Index            =   1
      End
      Begin VB.Menu mnuModeItem
         Caption          =   "Set Rect &2"
         Index            =   2
      End
   End
End
Attribute VB_Name = "RectAPI"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Option Explicit
'
' Project name  : RectAPI.vbp
' Program Name  : Windows ...Rect API Functions Demonstration
' VB/Win Module : RectAPI.frm
' DLL filename  : -
' Copyright     : (c)1997 by Sutthisak Phongthanapanich
'

' 32 bits API function declarations
Private Type Rect
    Left    As Long
    Top     As Long
    Right   As Long
    Bottom  As Long
End Type

Private Type PointAPI
    X   As Long
    Y   As Long
End Type

Private Declare Function InflateRect Lib "user32" (lpRect As Rect, _
        ByVal X As Long, ByVal Y As Long) As Long
Private Declare Function IntersectRect Lib "user32" (lpDestRect As Rect, _
        lpSrc1Rect As Rect, lpSrc2Rect As Rect) As Long
Private Declare Function IsRectEmpty Lib "user32" (lpRect As Rect) As Long
Private Declare Function OffsetRect Lib "user32" (lpRect As Rect, _
        ByVal X As Long, ByVal Y As Long) As Long
Private Declare Function PtInRect Lib "user32" (lpRect As Rect, _
        ByVal ptx As Long, ByVal pty As Long) As Long
Private Declare Function SetRect Lib "user32" (lpRect As Rect, _
        ByVal X1 As Long, ByVal Y1 As Long, ByVal X2 As Long, ByVal Y2 As Long) As Long
Private Declare Function SetRectEmpty Lib "user32" (lpRect As Rect) As Long
Private Declare Function SubtractRect Lib "user32" (lprcDst As Rect, _
        lprcSrc1 As Rect, lprcSrc2 As Rect) As Long
Private Declare Function UnionRect Lib "user32" (lpDestRect As Rect, _
        lpSrc1Rect As Rect, lpSrc2Rect As Rect) As Long

' Private rectangle variables
Private Rect1 As Rect
Private Rect2 As Rect
Private RectIntersect As Rect
Private RectSubtract As Rect
Private RectUnion As Rect

' Indicates the tracking
Private Captured%
Private ModeState%          ' 0 = Point detect mode
                            ' 1 = Setting rect1
                            ' 2 = Setting rect2
Private StartPoint As PointAPI
```

→

ลิสติงที่ 2 (ต่อ)

```
Private EndPoint As PointAPI

'
'   Displays information about the point
'
Private Sub ShowPointInfo()
    Dim p$
    Dim Xval&, Yval&

    ' read Point location
    Xval& = EndPoint.X
    Yval& = EndPoint.Y

    ' check Point location for each case
    If PtInRect(Rect1, Xval&, Yval&) Then p$ = "[R1] "
    If PtInRect(Rect2, Xval&, Yval&) Then p$ = p$ & "[R2] "
    If PtInRect(RectIntersect, Xval&, Yval&) Then p$ = p$ & "[RI] "
    If PtInRect(RectSubtract, Xval&, Yval&) Then p$ = p$ & "[RS] "
    If PtInRect(RectUnion, Xval&, Yval&) Then p$ = p$ & "[RU] "
    If p$ = "" Then p$ = "Not in any rectangle"

    Label1.Caption = p$
End Sub

Private Sub Check1_Click(Index As Integer)
    Picture1.Refresh
End Sub

'
' set active Rectangle for Sizing, Positioning and Emptyness
'
Private Sub Command1_Click(Index As Integer)
    Dim tmpRect As Rect
    Dim Success&
    Dim Xval&, Yval&

    If Option3(0).Value Then     ' Rect1
        Label1.Caption = "Rectangle 1 is active"
    Else                         ' Rect2
        Label1.Caption = "Rectangle 2 is active"
    End If

    Select Case Index
    Case 0   ' Up
        Xval& = 0: Yval& = -2
    Case 1   ' Down
        Xval& = 0: Yval& = 2
    Case 2   ' Left
        Xval& = -2: Yval& = 0
    Case 3   ' Right
        Xval& = 2: Yval& = 0
    End Select

    ' set Rect1 or Rect2 to tmpRect for smart coding
    If Option3(0).Value Then
        tmpRect = Rect1
    Else
        tmpRect = Rect2
    End If
    If Option2(0).Value Then     ' Size
        Success& = InflateRect(tmpRect, Xval&, Yval&)
    Else                         ' Position
        Success& = OffsetRect(tmpRect, Xval&, Yval&)
    End If
    ' set tmpRect back to Rect1 or Rect2
    If Option3(0).Value Then
        Rect1 = tmpRect
    Else
        Rect2 = tmpRect
    End If
    Picture1.Refresh
End Sub
```

→

ลิสติงที่ 2 (ต่อ)

```
'
' Empty active rectangle
'
Private Sub Command2_Click()
    Dim Success&

    If Option3(0).Value Then     ' Rect1
        Label1.Caption = "Empty Rectangle 1"
        Success& = SetRectEmpty(Rect1)
    Else                         ' Rect2
        Label1.Caption = "Empty Rectangle 2"
        Success& = SetRectEmpty(Rect2)
    End If
    Picture1.Refresh
End Sub

Private Sub Form_Load()
    ModeState% = 1    ' Set the initial value
End Sub

'
' set captured mode
'
Private Sub mnuModeItem_Click(Index As Integer)
    Dim Success&

    ModeState% = Index

    Select Case Index
    Case 0: Label1.Caption = "Set Point Indication"
    Case 1
        Label1.Caption = "Set Rectangle 1"
        ' Empty Rect1 if exist
        If IsRectEmpty(Rect1) = 0 Then
            Success& = SetRectEmpty(Rect1)
        End If
    Case 2
        Label1.Caption = "Set Rectangle 2"
        ' Empty Rect2 if exist
        If IsRectEmpty(Rect2) = 0 Then
            Success& = SetRectEmpty(Rect2)
        End If
    End Select

    Picture1.Refresh
End Sub

'
' Manipulate logical operation
'
Private Sub Option1_Click(Index As Integer)
    Select Case Index
    Case 0: Label1.Caption = "Intersection area"
    Case 1: Label1.Caption = "Subtract area (R1 - R2)"
    Case 2: Label1.Caption = "Subtract area (R2 - R1)"
    Case 3: Label1.Caption = "Union area"
    Case 4: Label1.Caption = "None"
    End Select

    Picture1.Refresh
End Sub

Private Sub Option2_Click(Index As Integer)
    If Index = 0 Then
        Command1(0).Caption = "-Y"
        Command1(1).Caption = "+Y"
        Command1(2).Caption = "-X"
        Command1(3).Caption = "+X"
    Else
        Command1(0).Caption = "&U"
```

→

ลิสติงที่ 2 (ต่อ)

```
        Command1(1).Caption = "&D"
        Command1(2).Caption = "&L"
        Command1(3).Caption = "&R"
    End If
End Sub

Private Sub Option3_Click(Index As Integer)
    Select Case Index
    Case 0: Label1.Caption = "Rectangle 1 is active"
    Case 1: Label1.Caption = "Rectangle 2 is active"
    End Select
End Sub

'
' Record current mouse location in StartPoint, EndPoin
'
Private Sub Picture1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y
As Single)
    StartPoint.X = X
    StartPoint.Y = Y
    EndPoint.X = X
    EndPoint.Y = Y

    ' Drawing will be Xor operation
    Picture1.DrawMode = vbNotXorPen
    Captured% = -1
End Sub

'
' Erase the prior rectangle and draw new location
'
Private Sub Picture1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y
As Single)
    If ModeState% <> 0 And Captured% Then
        Picture1.Line (StartPoint.X, StartPoint.Y)-(EndPoint.X, EndPoint.Y), , B
        Picture1.Line (StartPoint.X, StartPoint.Y)-(X, Y), , B
    End If

    EndPoint.X = X
    EndPoint.Y = Y
End Sub

'
' Erase prior rectangle and save currently information
'
Private Sub Picture1_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As
Single)
    ' If we're not mouse tracking, exit the subroutine
    If Not Captured% Then Exit Sub

    If ModeState% <> 0 Then
        Picture1.Line (StartPoint.X, StartPoint.Y)-(EndPoint.X, EndPoint.Y), , B
    End If
    EndPoint.X = X
    EndPoint.Y = Y

    Select Case ModeState%
    Case 0: ShowPointInfo ' Show point information
    Case 1: SetRect Rect1, StartPoint.X, StartPoint.Y, EndPoint.X, EndPoint.Y
    Case 2: SetRect Rect2, StartPoint.X, StartPoint.Y, EndPoint.X, EndPoint.Y
    End Select
    Captured% = 0

    ' Restore the original drawing mode
    Picture1.DrawMode = vbCopyPen
    Picture1.Refresh
End Sub

'
' Redraw for all rectangle
'
```

→

ลิสติงที่ 2 (ต่อ)

```
Private Sub Picture1_Paint()
    Dim Success&

    ' View Rect1
    If Check1(0).Value Then
        Picture1.Line (Rect1.Left, Rect1.Top)-(Rect1.Right, Rect1.Bottom), QBColor(14), B
    End If
    ' View Rect1
    If Check1(1).Value Then
        Picture1.Line (Rect2.Left, Rect2.Top)-(Rect2.Right, Rect2.Bottom), QBColor(12), B
    End If

    If Option1(0).Value Then        ' Intersection rectangles
        Success& = IntersectRect(RectIntersect, Rect1, Rect2)
        Picture1.Line (RectIntersect.Left, RectIntersect.Top)-(RectIntersect.Right, _
                       RectIntersect.Bottom), QBColor(8), BF
    ElseIf Option1(1).Value Then    ' Subtraction rectangles R1 - R2
        Success& = SubtractRect(RectSubtract, Rect1, Rect2)
        Picture1.Line (RectSubtract.Left, RectSubtract.Top)-(RectSubtract.Right, _
                       RectSubtract.Bottom), QBColor(4), BF
    ElseIf Option1(2).Value Then    ' Subtraction rectangles R2 - R1
        Success& = SubtractRect(RectSubtract, Rect2, Rect1)
        Picture1.Line (RectSubtract.Left, RectSubtract.Top)-(RectSubtract.Right, _
                       RectSubtract.Bottom), QBColor(4), BF
    ElseIf Option1(3).Value Then    ' Union rectangles
        Success& = UnionRect(RectUnion, Rect1, Rect2)
        Picture1.Line (RectUnion.Left, RectUnion.Top)-(RectUnion.Right, _
                       RectUnion.Bottom), QBColor(5), BF
    Else                            ' None
        Success& = SetRectEmpty(RectIntersect)
        Success& = SetRectEmpty(RectSubtract)
        Success& = SetRectEmpty(RectUnion)
        Picture1.Line (RectIntersect.Left, RectIntersect.Top)-(RectIntersect.Right, _
                       RectIntersect.Bottom), QBColor(8), BF
        Picture1.Line (RectSubtract.Left, RectSubtract.Top)-(RectSubtract.Right, _
                       RectSubtract.Bottom), QBColor(4), BF
        Picture1.Line (RectUnion.Left, RectUnion.Top)-(RectUnion.Right, _
                       RectUnion.Bottom), QBColor(5), BF
    End If
End Sub
```

■

เทคนิคประจำคอลัมน์

ถาม สร้างแผ่น setup แอพพลิเคชันที่เขียนด้วย Visual Basic 4.0 ด้วย Setup Wizard ปรากฏว่าเกิดข้อผิดพลาดไม่มีไฟล์ MSVCRT.DLL จะทำอย่างไร?

ตอบ อาจจะเป็นเพราะว่าไฟล์ไลบรารี OLEPRO32.DLL ที่มากับ Visual Basic 4.0 นั้นมีความต้องการใช้บางฟังก์ชันในไฟล์ไลบรารี MSVCRT.DLL ซึ่งเป็นไฟล์รันไทม์ของ Visual C++ 4.0 แต่ผู้อ่านอาจจะมีการติดตั้ง Visual C++ 4.2 หรือ Windows NT 4.0 ซึ่งจะมาพร้อมกับไฟล์ไลบรารี MSVCRT.DLL เวอร์ชันที่ใหม่กว่า ดังนั้น Setup Wizard ของ Visual Basic 4.0 จึงไม่สามารถที่จะติดตั้ง MSVCRT.DLL นั่นเอง แต่สำหรับผู้อ่านที่มีการติดตั้ง Microsoft Office 97 หรือ Internet Explorer 3.0 บนวินโดวส์ 95 ก็ไม่ต้องกังวลเรื่องนี้ครับ เพราะแอพพลิเคชันทั้งสองจะมาพร้อมกับไฟล์ไลบรารี OLEPRO32.DLL และ MSVCRT.DLL ซึ่งเป็นเวอร์ชันใหม่ทั้งคู่ ดังนั้นจึงไม่มีปัญหาเรื่องการติดตั้ง